Diffusion Model and ODE/Flows

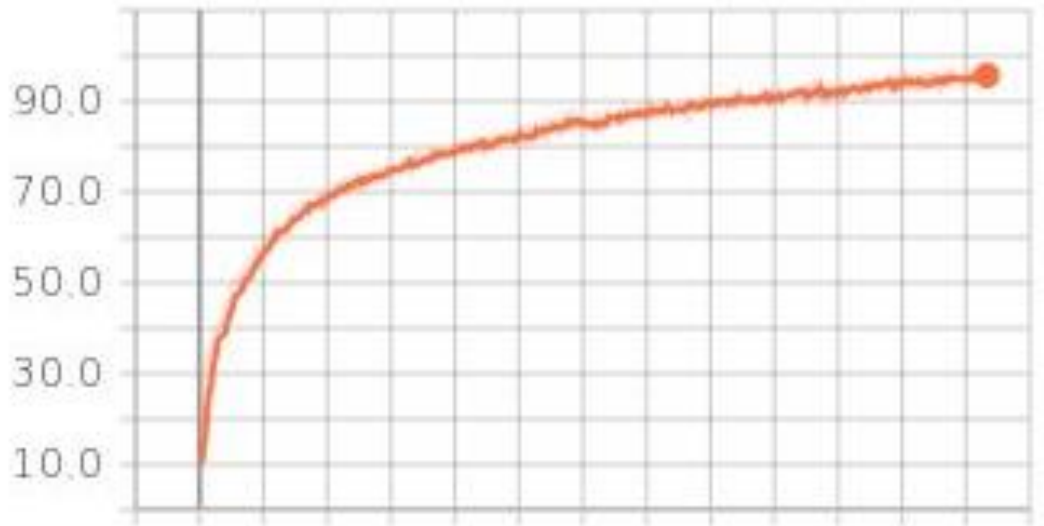
Lecture 10

18-789

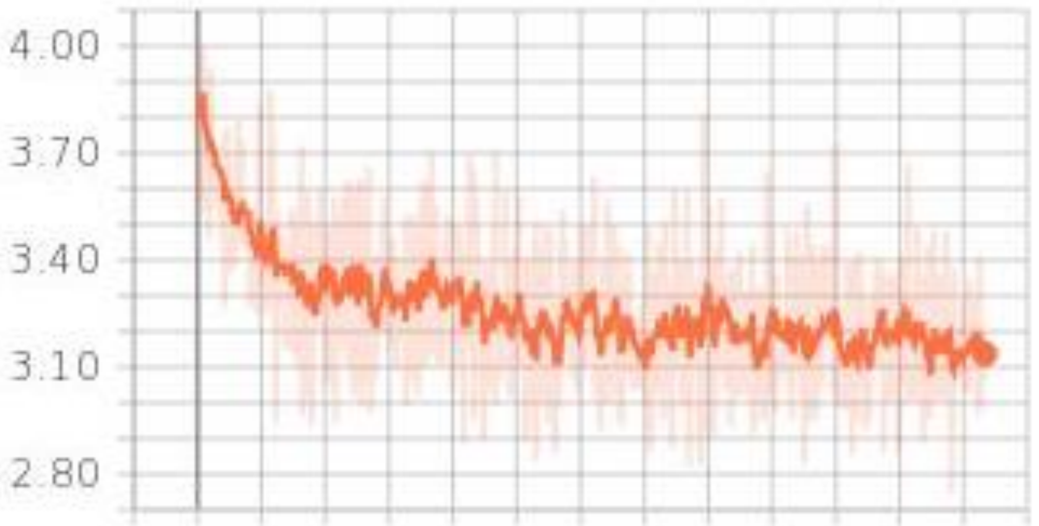
Common Questions in Homework

- VAE loss functions

KL Loss



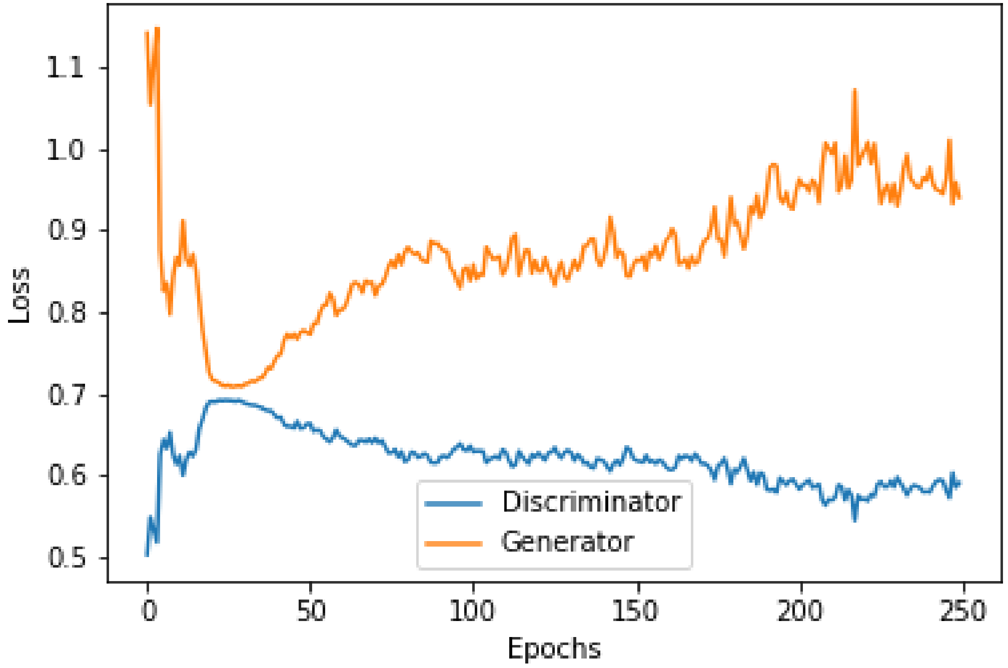
Reconstruction Loss

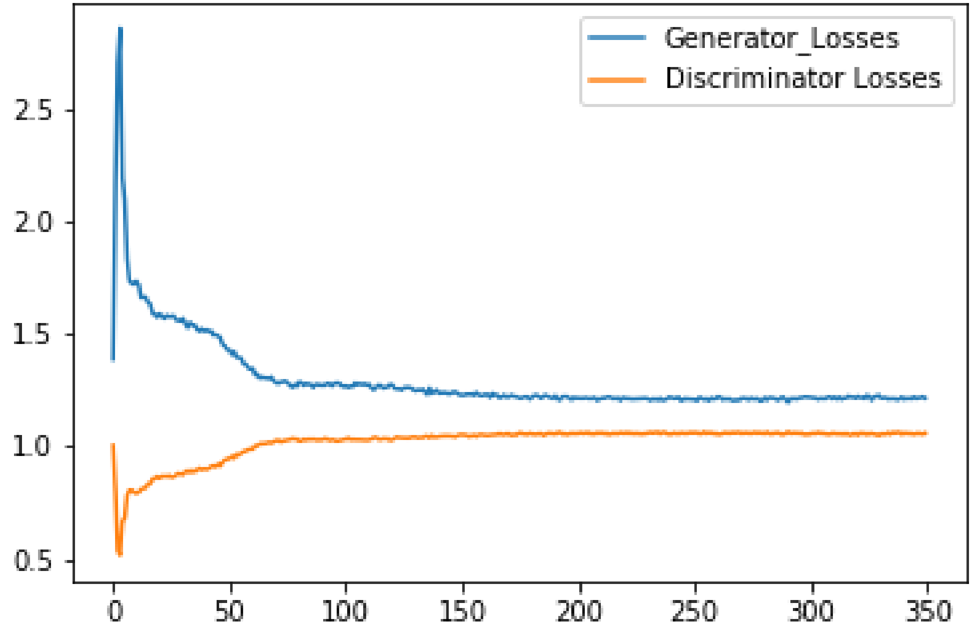


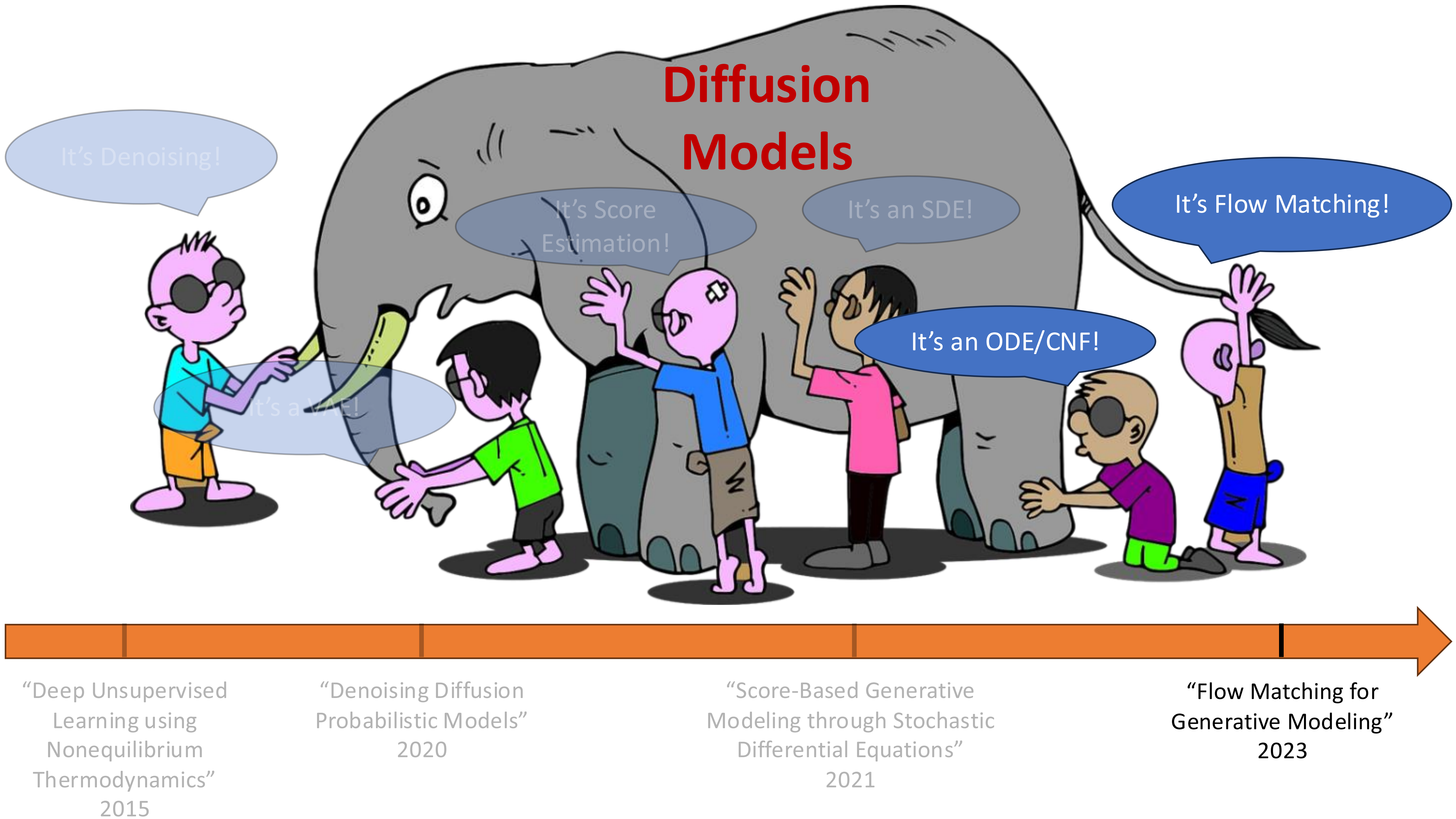
Common Questions in Homework

- (Vanilla) GAN loss functions
  - They are not very informative!

Both could work…

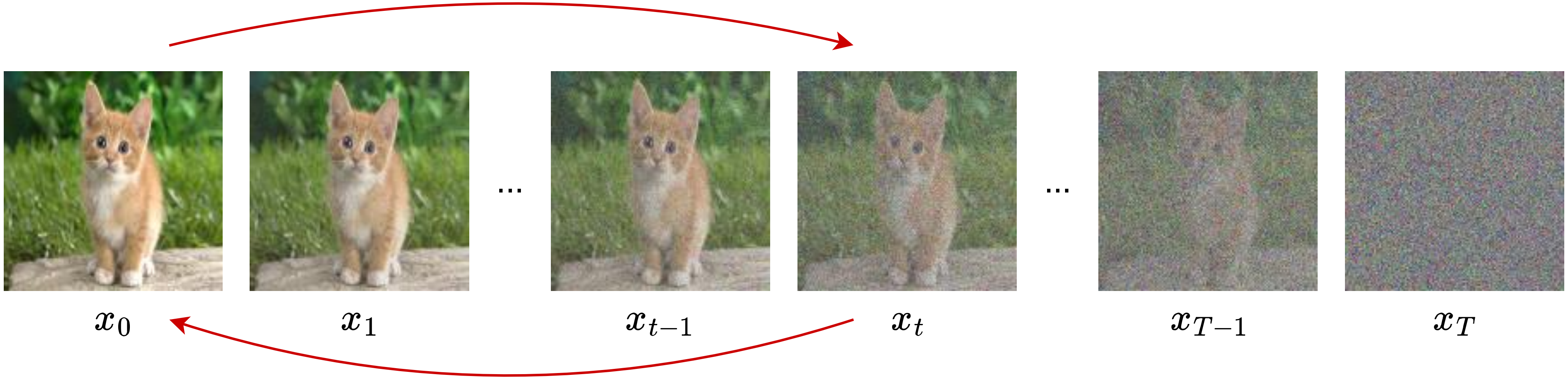





Diffusion Models
It’s Denoising!
It’s Score Estimation!
It’s an SDE!
It’s Flow Matching!
It’s an ODE/CNF!
It’s a VAE!
“Deep Unsupervised Learning using Nonequilibrium Thermodynamics” 2015
“Denoising Diffusion Probabilistic Models” 2020
“Score-Based Generative Modeling through Stochastic Differential Equations” 2021
“Flow Matching for Generative Modeling” 2023

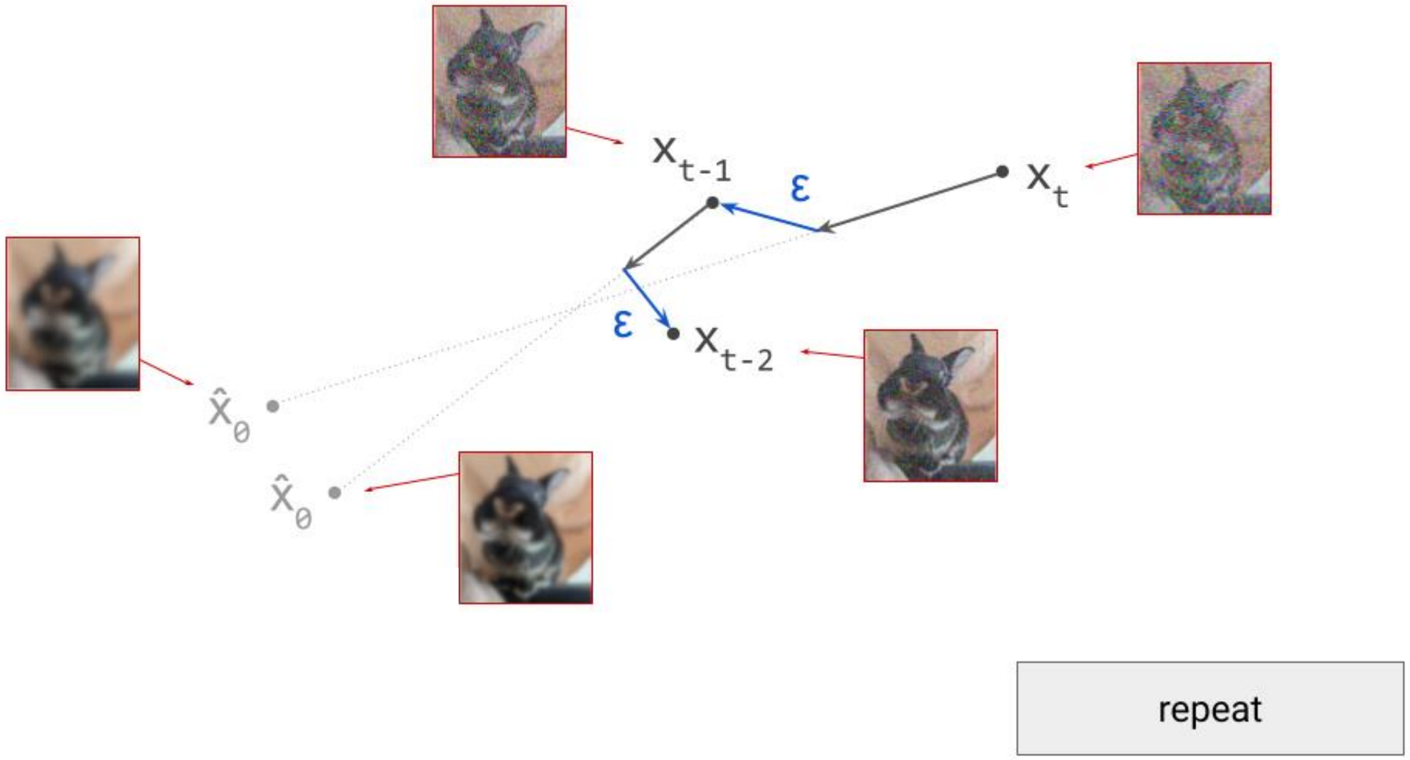
Recap: Diffusion Models

- Training: Denoising objective



Recap: Diffusion Models

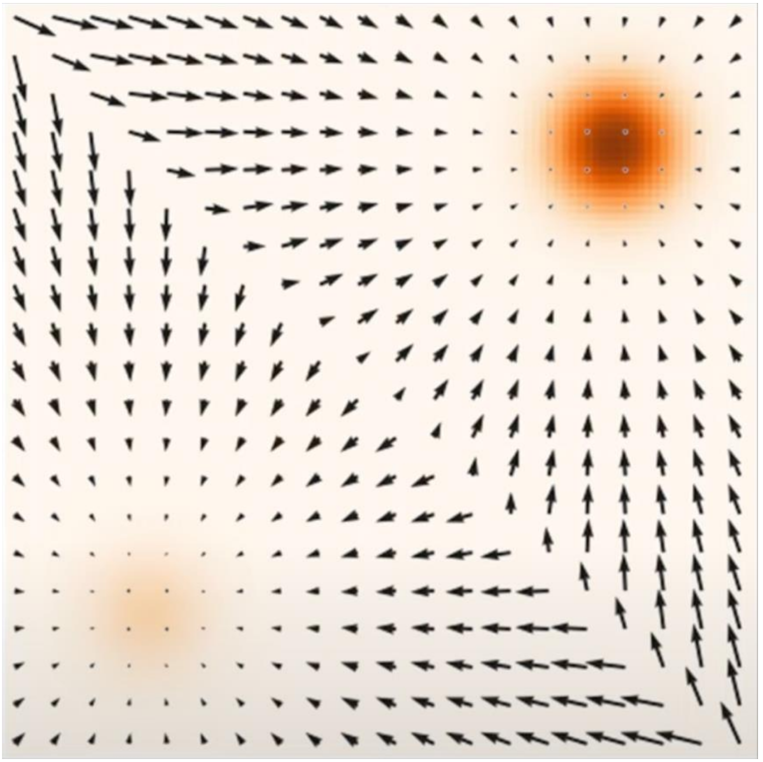
- Training: Denoising objective
- Inference: Iterative denoising



Recap: Diffusion Models

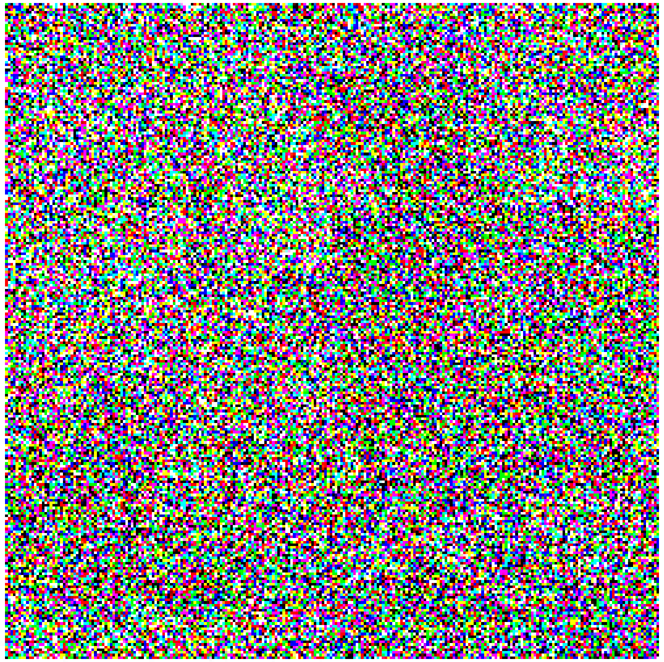
- Training: Denoising score matching

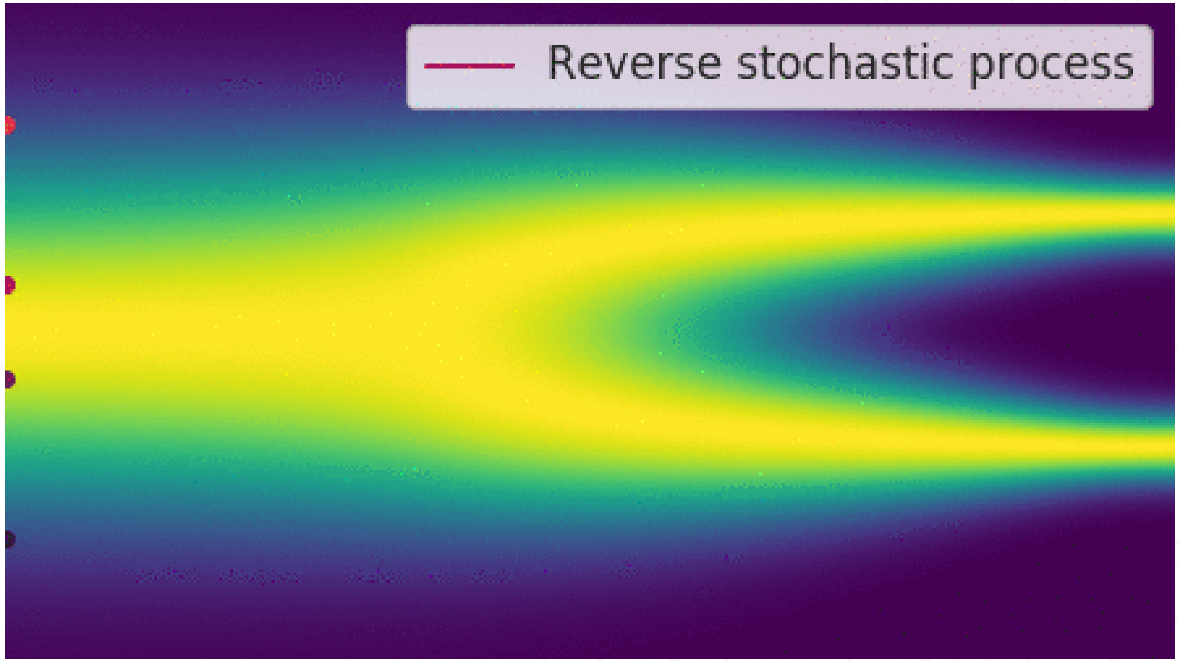
$$s_\theta(x;t) = \nabla_x \log p_t(x)$$



Recap: Diffusion Models

- Training: Denoising score matching
- Inference: Solve an SDE



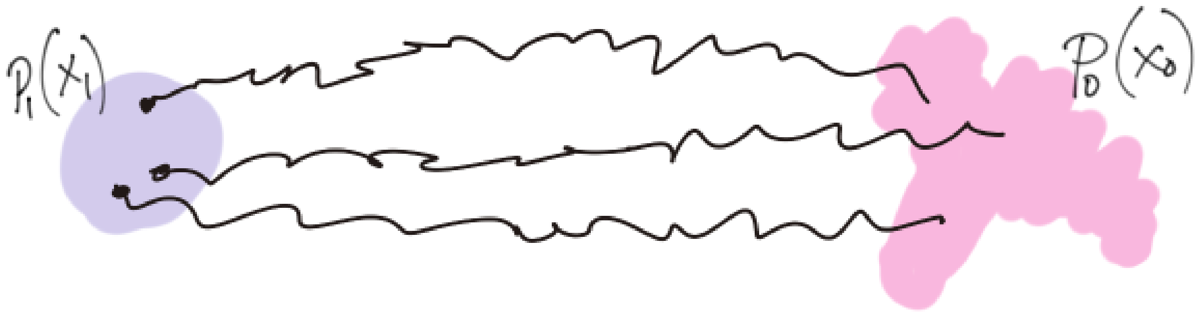


$$d\boldsymbol{x} = (f(\boldsymbol{x}, t) - g(t)^2 s_\theta(x; t))dt + g(t)d\bar{\boldsymbol{w}}$$

Diffusion Model is an ODE

- Every SDE has a corresponding "probability flow" ODE that has the same marginal distribution at all time t.

SDE: $d\boldsymbol{x} = (f(\boldsymbol{x}, t) - g(t)^2 \nabla_x \log p_t(x))dt + g(t)d\overline{\boldsymbol{w}}$



ODE: $d\boldsymbol{x} = (f(\boldsymbol{x}, t) - \frac{1}{2} g(t)^2 \nabla_x \log p_t(x))dt$

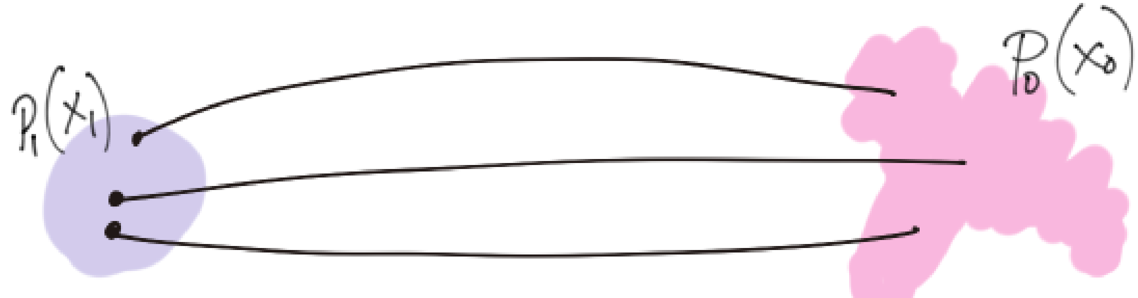


Figure source: "Match flows, not scores", Ayan Das

ODE: a deterministic interpretation

- The forward process determines a vector field/flow that connect Gaussian and data distribution. (ODE = flow = vector field = velocity)
- Training: estimate the vector field/tangents of flows.

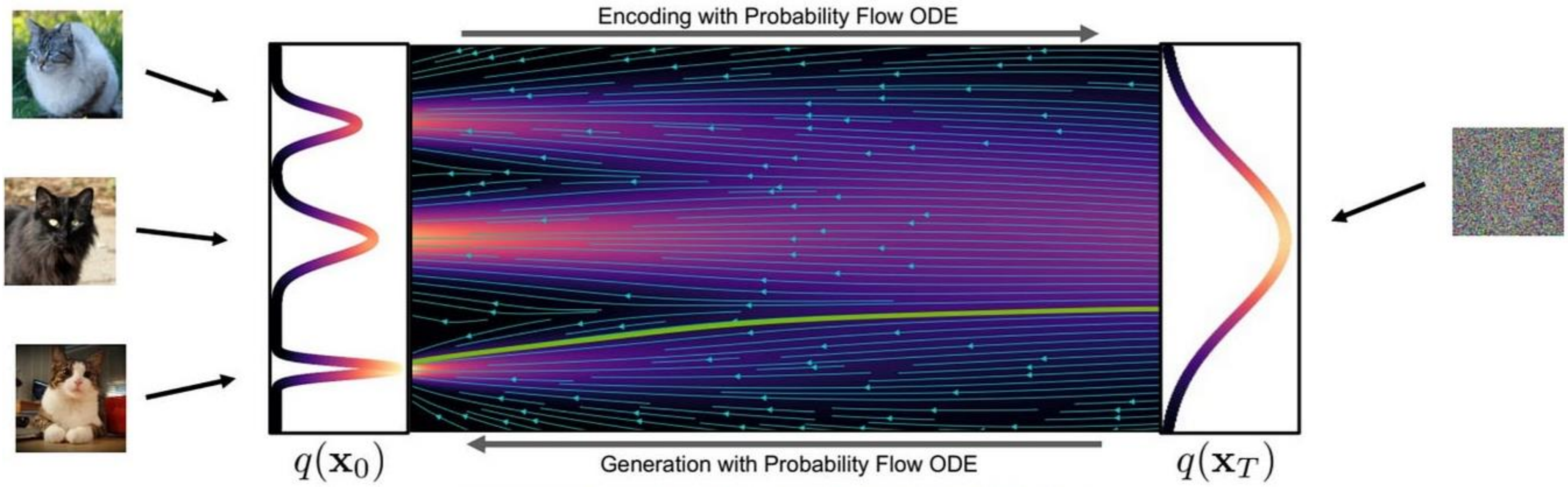
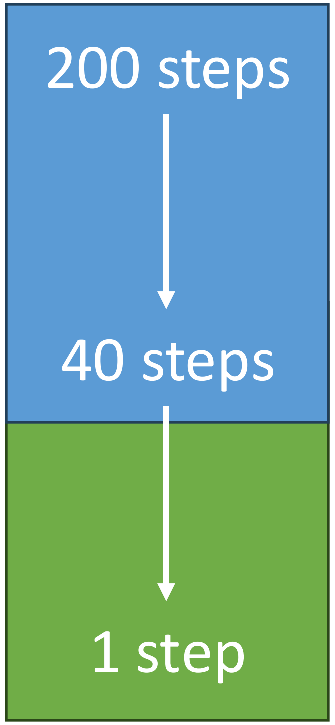


Figure source: “GENIE: Higher-Order Denoising Diffusion Solvers”

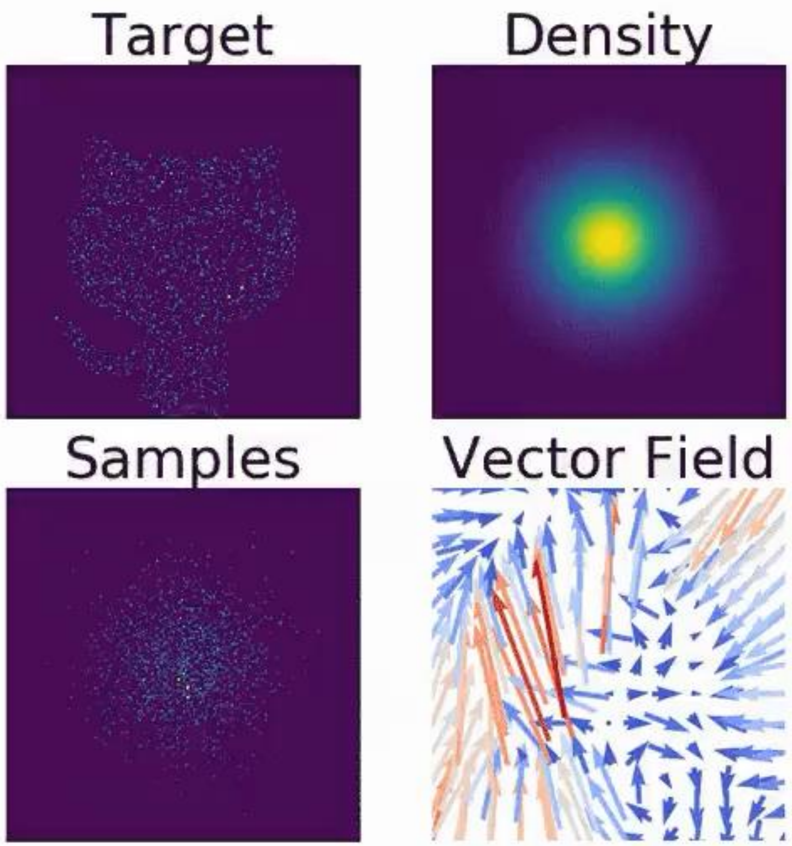
ODE: a deterministic interpretation

- The forward process determines a vector field/flow that connect Gaussian and data distribution. (ODE = flow = vector field = velocity)
- Training: estimate the vector field/tangents of flows.
- Inference: solve an ODE
  - DDIM [Song 2020]
  - Exponential Integrator [Zhang 2022]
  - DPM-Solver [Lu 2022]
  - Open the door to distillation – sampling in **one** step!



Remember: Continuous Normalizing Flows

- Sampling is the same (solving an ODE defined by a neural network)
- Training of CNF is slow:
  - $\log p(x_0) = \log p(x_1) + \int_0^1 \mathrm{Trace}\left(\frac{\partial f}{\partial z_t}\right) dt$
  - $x_1 = x_0 + \int_0^1 f(x_t, t)\, dt$ $\quad \int$ is slow
- Training Diffusion is much more efficient



ODE allows exact likelihood evaluation

- $\log p(x_0) = \log p(x_1) + \int_0^1 \mathrm{Trace}\left(\frac{\partial f}{\partial z_t}\right) dt$

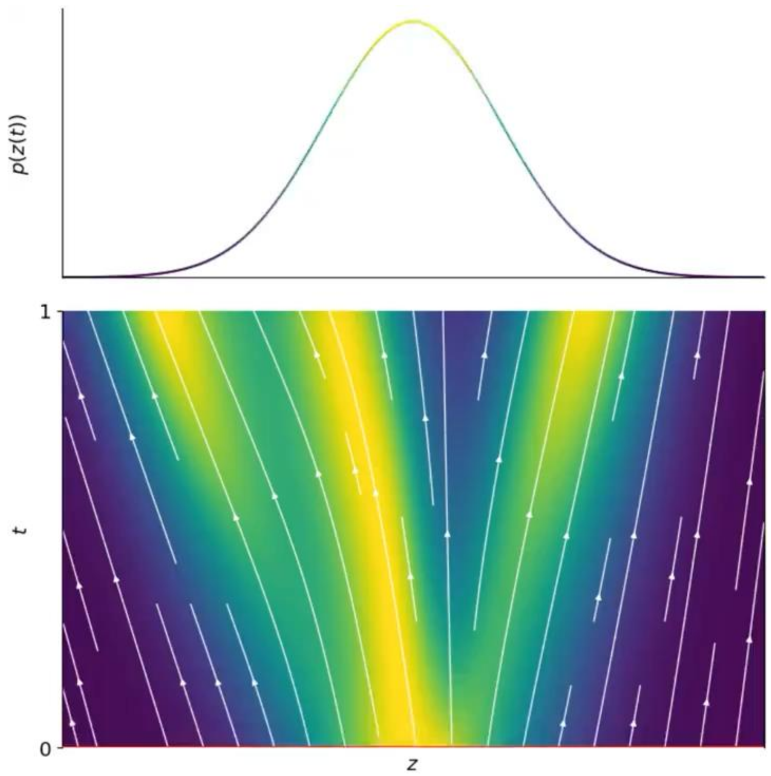


Table 2: NLLs and FIDs (ODE) on CIFAR-10.

	Model	NLL Test ↓	FID ↓
Normalizing Flows	RealNVP (Dinh et al., 2016)	3.49	-
	iResNet (Behrmann et al., 2019)	3.45	-
	Glow (Kingma & Dhariwal, 2018)	3.35	-
	MintNet (Song et al., 2019b)	3.32	-
	Residual Flow (Chen et al., 2019)	3.28	46.37
	FFJORD (Grathwohl et al., 2018)	3.40	-
	Flow++ (Ho et al., 2019)	3.29	-
Diffusion (VAE interpretation)	DDPM (L) (Ho et al., 2020)	$\leqslant 3.70^*$	13.51
	DDPM (L_{simple}) (Ho et al., 2020)	$\leqslant 3.75^*$	3.17
Autoregressive Models	PixelCNN (Oord et al., 2016)	3.03	
	PixelCNN++ (Salimans et al., 2017)	2.92	
	Image Transformer (Parmar et al., 2018	2.90	
	PixelSNAIL (Chen et al., 2017)	2.85	
	Sparse Transformer 59M (strided)	**2.80**	
Diffusion (ODE interpretation)	DDPM	3.28	3.37
	DDPM cont. (VP)	3.21	3.69
	DDPM cont. (sub-VP)	3.05	3.56
	DDPM++ cont. (VP)	3.16	3.93
	DDPM++ cont. (sub-VP)	3.02	3.16
	DDPM++ cont. (deep, VP)	3.13	3.08
	DDPM++ cont. (deep, sub-VP)	**2.99**	**2.92**

Let's further simplify it!

$$\text{ODE: } dx_t = \underbrace{\left(f(\boldsymbol{x}, t) - \frac{1}{2} g(t)^2 \overbrace{\nabla_x \log p_t(x)}^{\text{Instead of learning this ...}}\right)}_{\text{... why not just learn this?}} dt$$

$f, g, \nabla_x \log p_t(x)$ all come from the SDE interpretation,
do we still need them if we only care about the corresponding ODE?

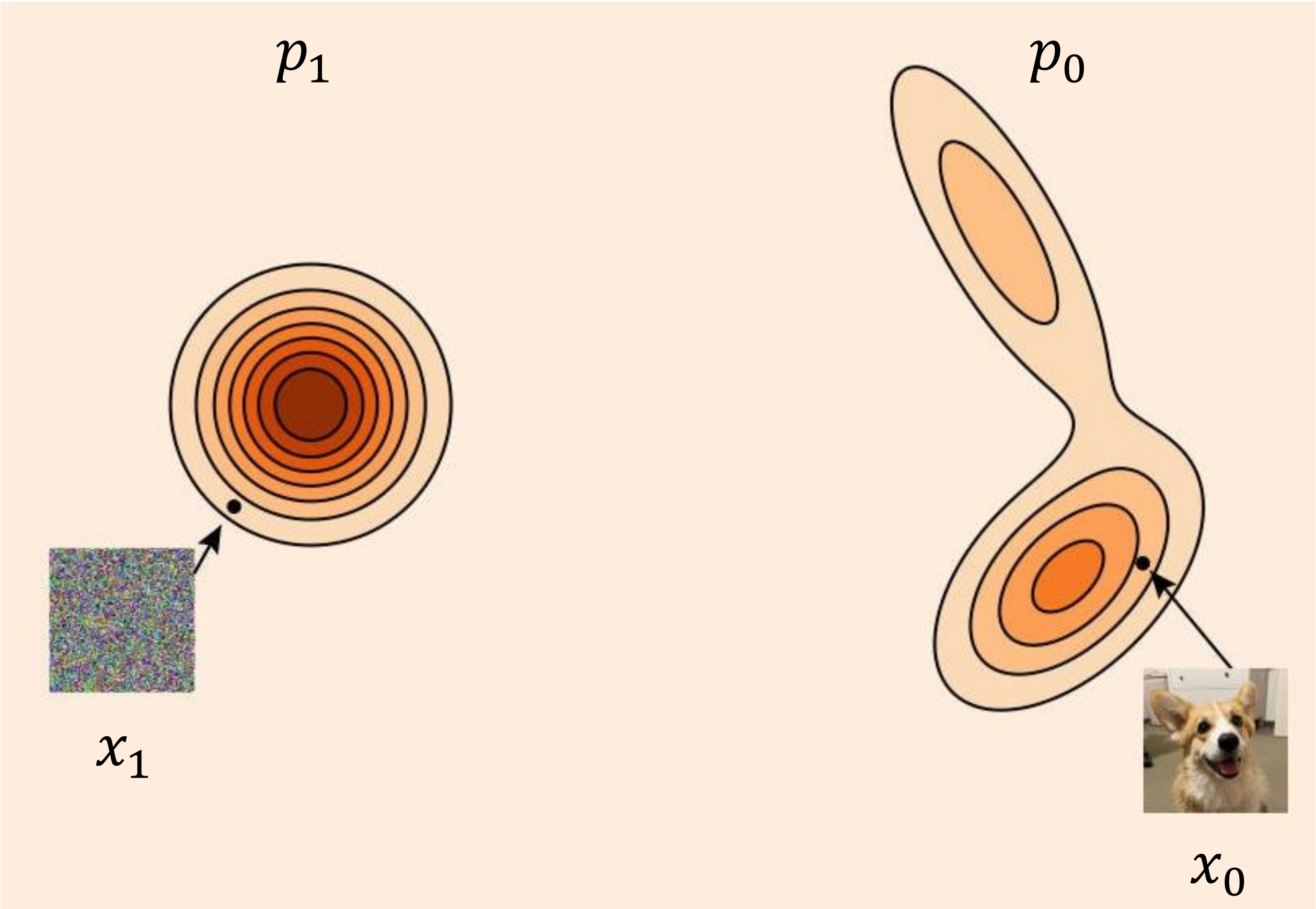
No!

Flow Matching: a simplified perspective

- Diffusion Models:
  - Predetermine a flow that transports noise to data (by setting f, g):
$$dx_t = (f(\boldsymbol{x}, t) - \frac{1}{2} g(t)^2 \nabla_x \log p_t(x))dt$$
  - Train a neural network to approximate the score: $s_\theta(x; t) \approx \nabla_x \log p_t(x)$
- Flow Matching:
  - Predetermine a flow $u_t(x)$ that transports noise to data:
$$dx_t = u_t(x)dt$$
  - Train a neural network to approximate the velocity: $v_t^\theta(x) \approx u_t(x)$

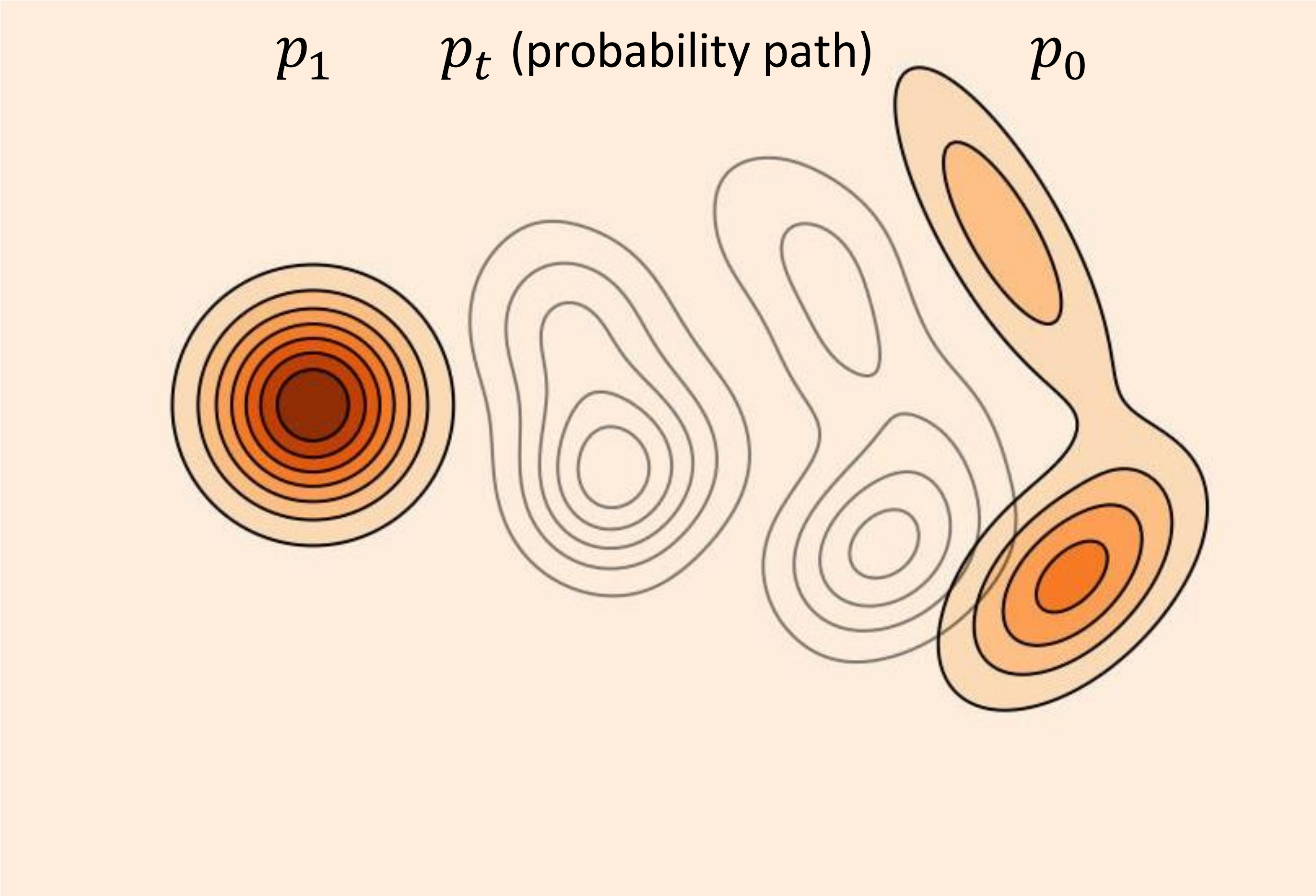
(flow/vector field)

Flow Matching



Lipman et al., "Flow Matching Guide and Code"

Flow Matching

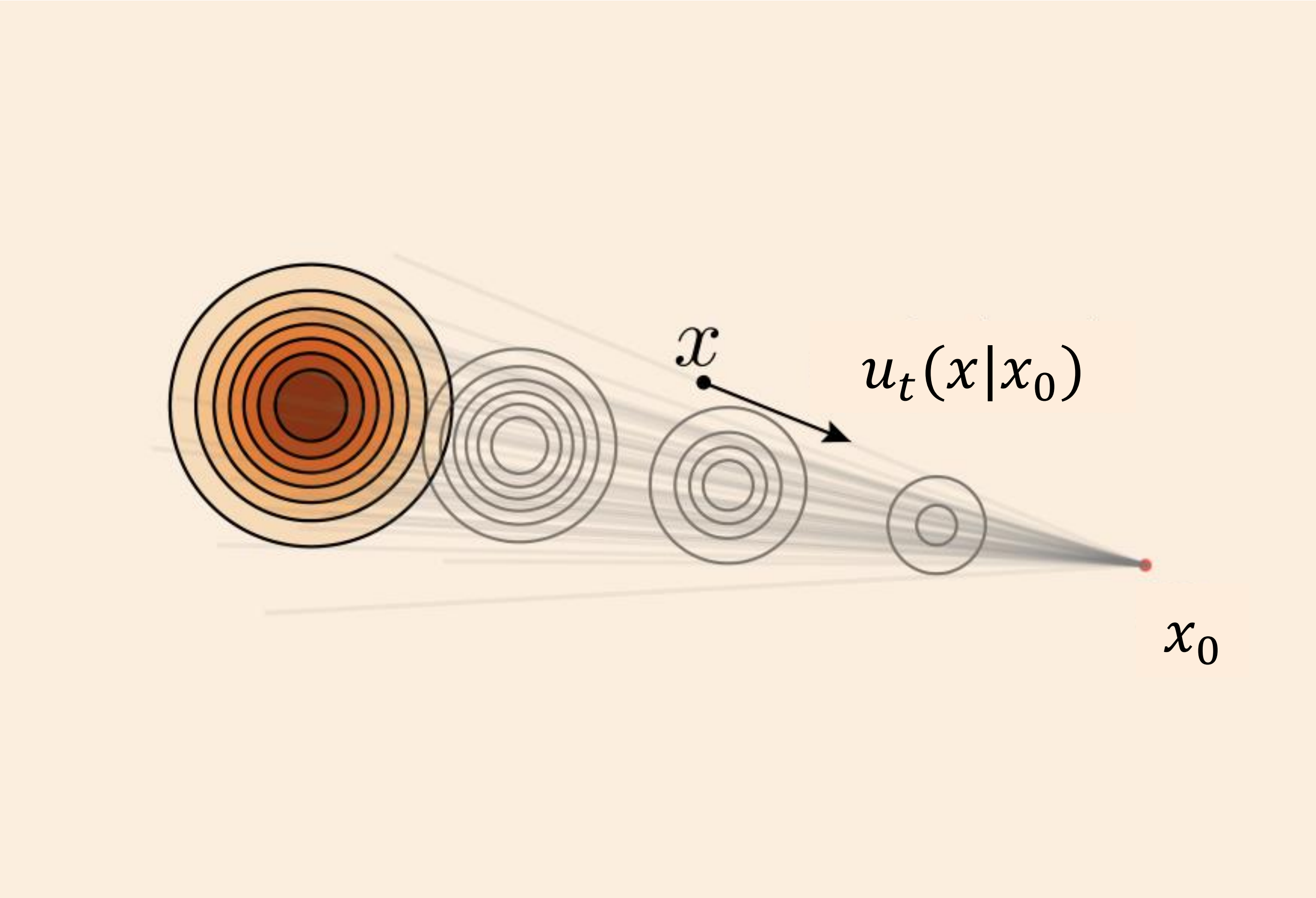


Given a vector field $u_t(x)$ that transports the density p_0 to p_1, we train a NN to regress the vector field:

$$\min_{\theta} \mathbb{E}_{t,\, x\sim p_t(x)} \left\| v_t^{\theta}(x) - u_t(x) \right\|^2$$

☹ We don't know $u_t(x)$

(Conditional) Flow Matching



Construct *conditional* vector fields that are known!
e.g., linear interpolation

$$\min_\theta \mathbb{E}_{t,\, x\sim p_t(x)} \left\| v_t^\theta(x) - u_t(x) \right\|^2$$

$$= \min_\theta \mathbb{E}_{t,\, x_0,\, x\sim p_t(x|x_0)} \left\| v_t^\theta(x) - u_t(x|x_0) \right\|^2$$

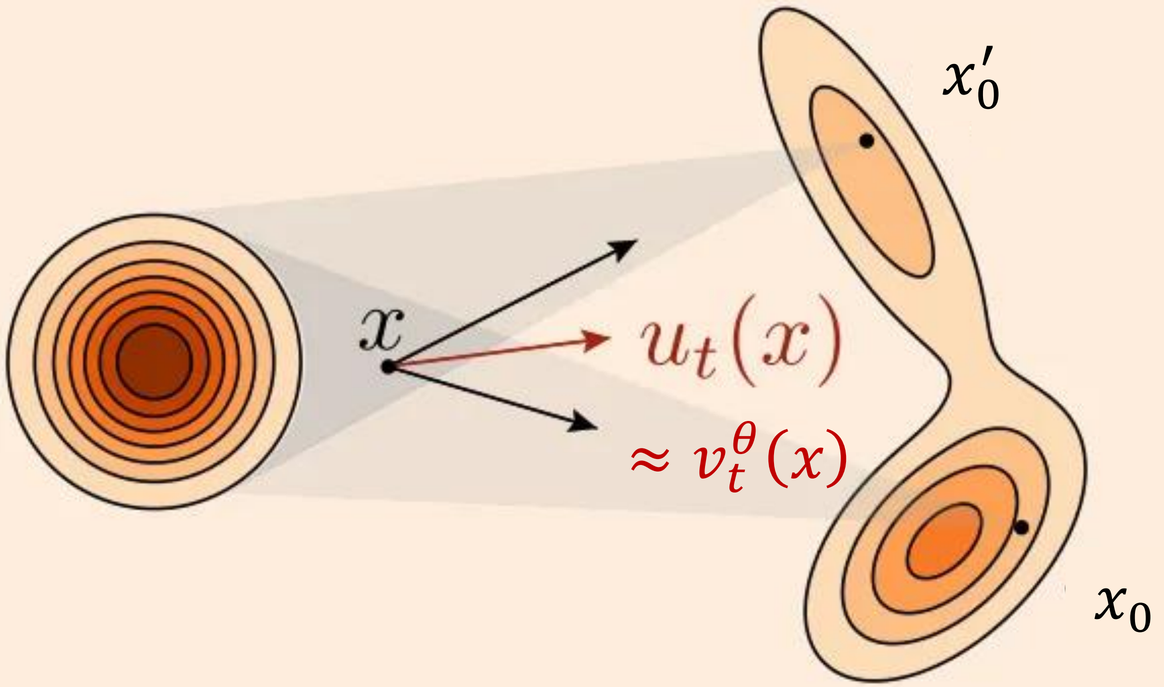
Denoising score matching

$$\mathbb{E}_{\tilde{x}\sim p(\tilde{x}|x),\, x\sim p(x)} \| s_\theta(\tilde{x}) - \nabla_{\tilde{x}} \log p(\tilde{x}) \|^2$$

$$= \mathbb{E}_{\tilde{x}\sim p(\tilde{x}|x),\, x\sim p(x)} \| s_\theta(\tilde{x}) - \nabla_{\tilde{x}} \log p(\tilde{x}|x) \|^2 + \mathrm{C}$$

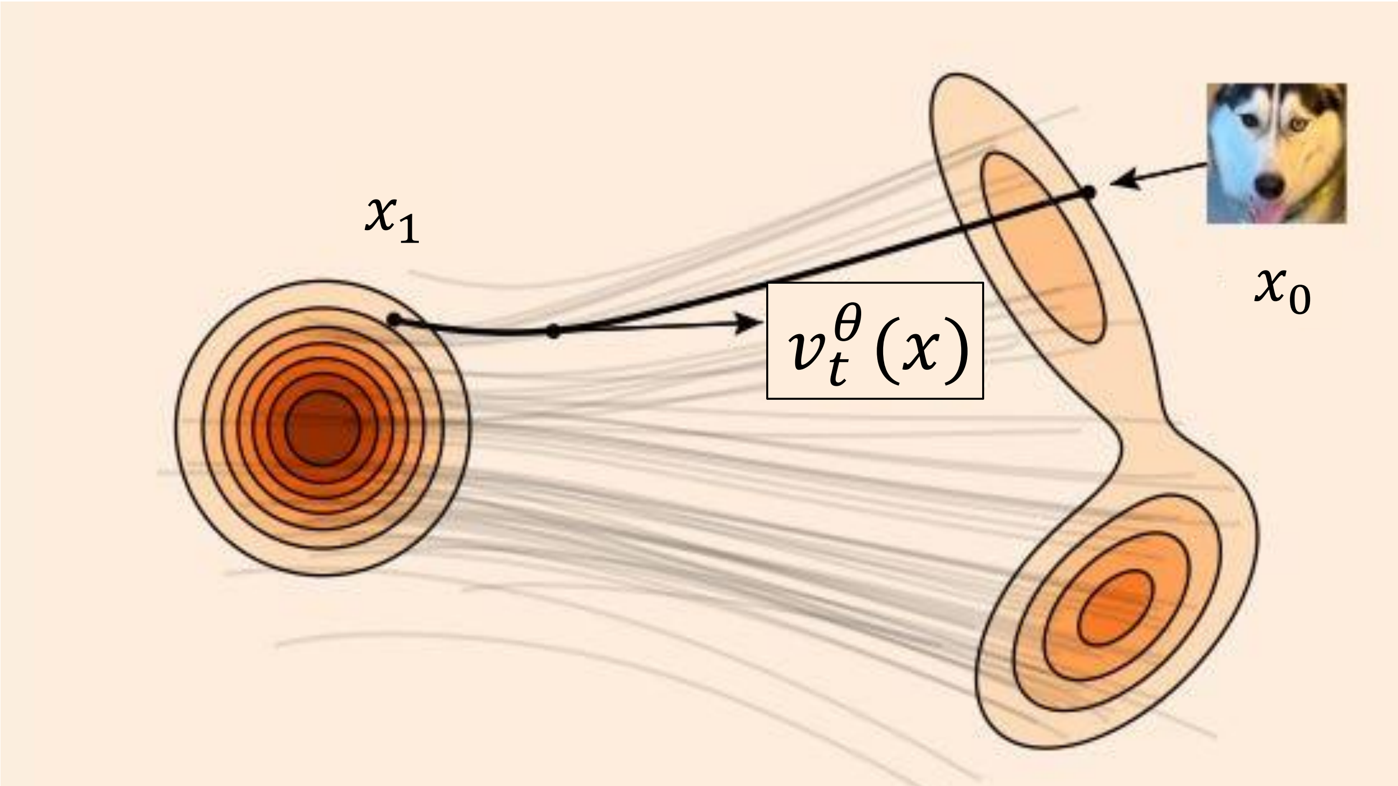
VAE derivation: $\log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})} \Rightarrow \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1}, x_0)}$

Marginalization



$$\min_\theta \mathbb{E}_{t,\, x_0,\, x\sim p_t(x|x_0)} \left\| v_t^\theta(x) - u_t(x|x_0) \right\|^2$$

Sampling by solving an ODE



Sample $x_1 \sim N(0, I)$

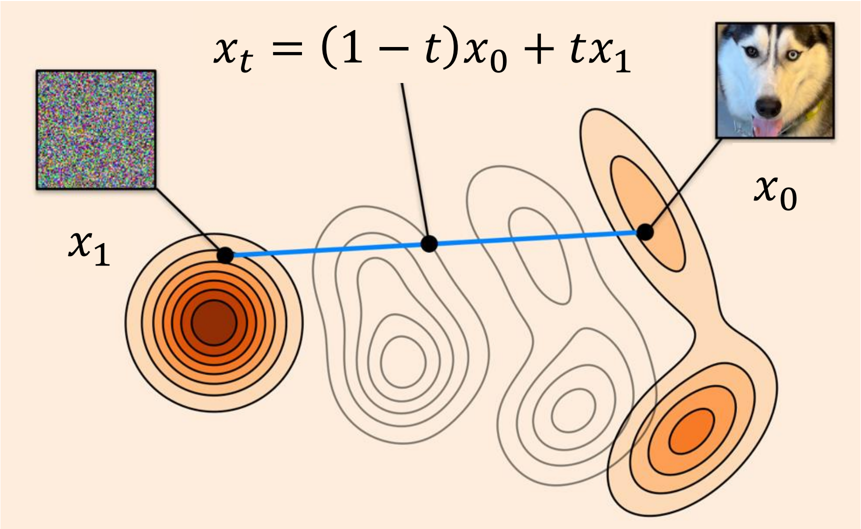
Solve ODE: $dx_t = v_t^\theta(x)dt$

Example: Linear Interpolation Flow Matching

- Sample $x_0 \sim p_{data}, x_1 \sim N(0, I), t \sim \text{Uniform}(0, 1)$
- Create $x_t = (1 - t)x_0 + tx_1$
- Target vector is $u_t(x|x_0) = x_0 - x_1$
- Train the model $v_t^\theta(x)$ with the regression loss:

$$\left\| v_t^\theta(x) - (x_0 - x_1) \right\|^2$$

Question: Is the target vector field $u_t(x)$ always straight?



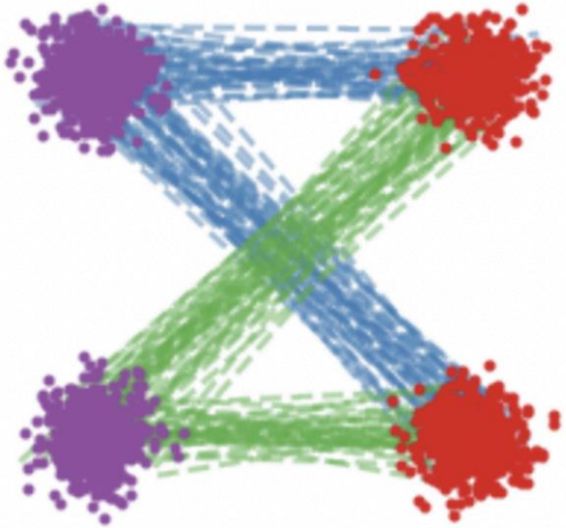
Diffusion is Flow Matching

- Create "noisy" data:
  - Flow matching: $x_t = (1-t)x_0 + t \cdot \epsilon$
  - Diffusion: $x_t = \sqrt{\alpha_t} x_0 + \sqrt{1-\alpha_t}\epsilon$
  - More generally: $x_t = \alpha_t x_0 + \sigma_t \epsilon$
- Train the model:
  - Flow matching: regressing the target $(x_0 - \epsilon)$
  - Diffusion: normally regressing ϵ, but remember
    - $\tilde{x}_{t-1}, x_0, \epsilon$ or any of their linear combinations are equivalent targets
    - Including $x_0 - \epsilon$!

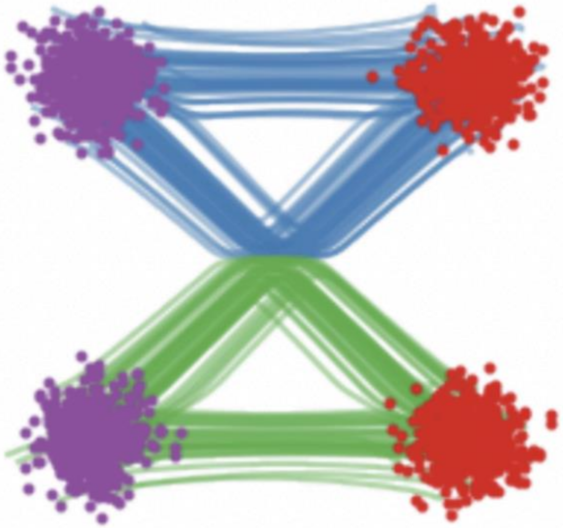
Question: What's the benefit of the Flow Matching interpretation?

Non-Gaussian Flow Matching

$p_0 = p_{\text{cat}}$



Training: sample $x_0 \sim p_0, x_1 \sim p_1$ independently and interpolate

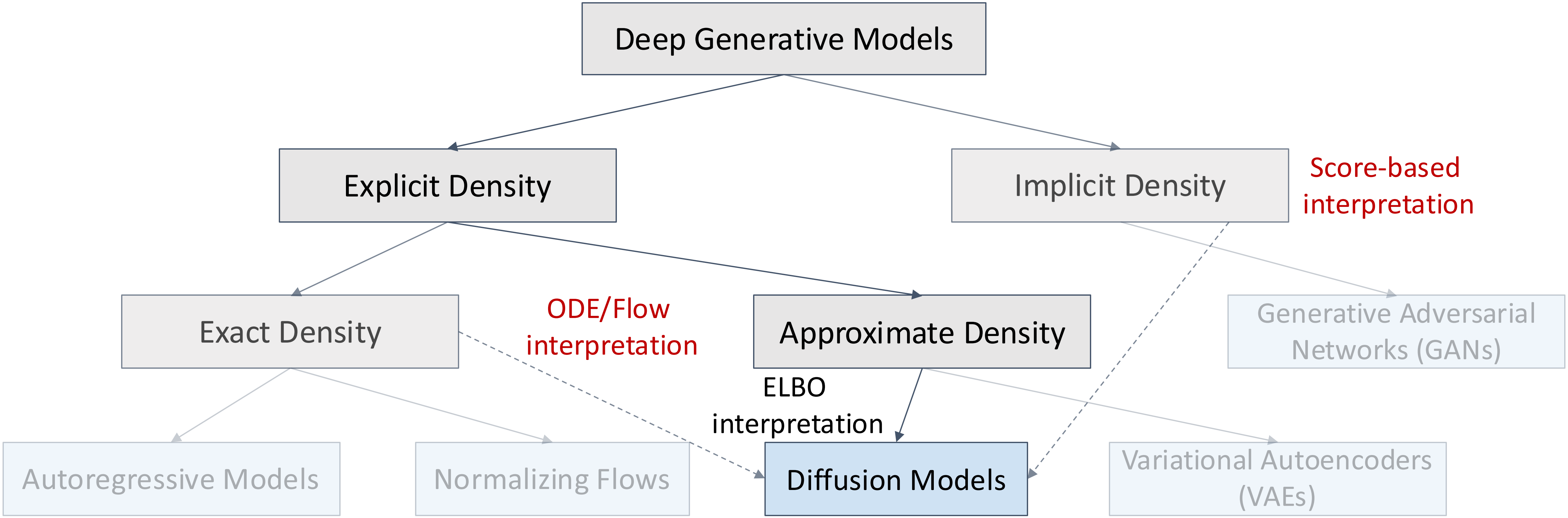


Learned Flow

$p_1 = p_{\text{cat}}$

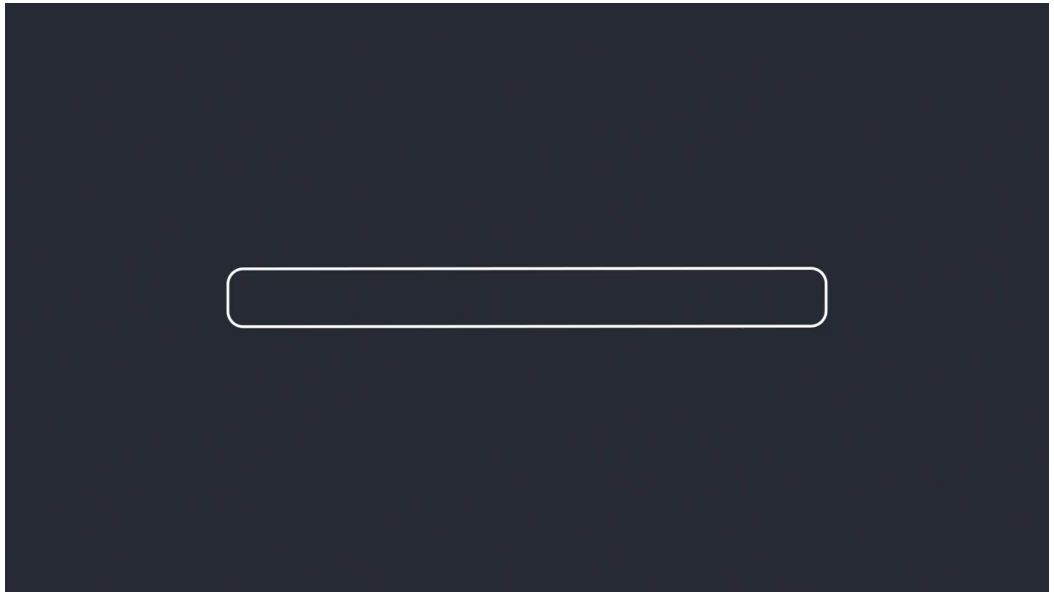
Liu et al., "Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow"

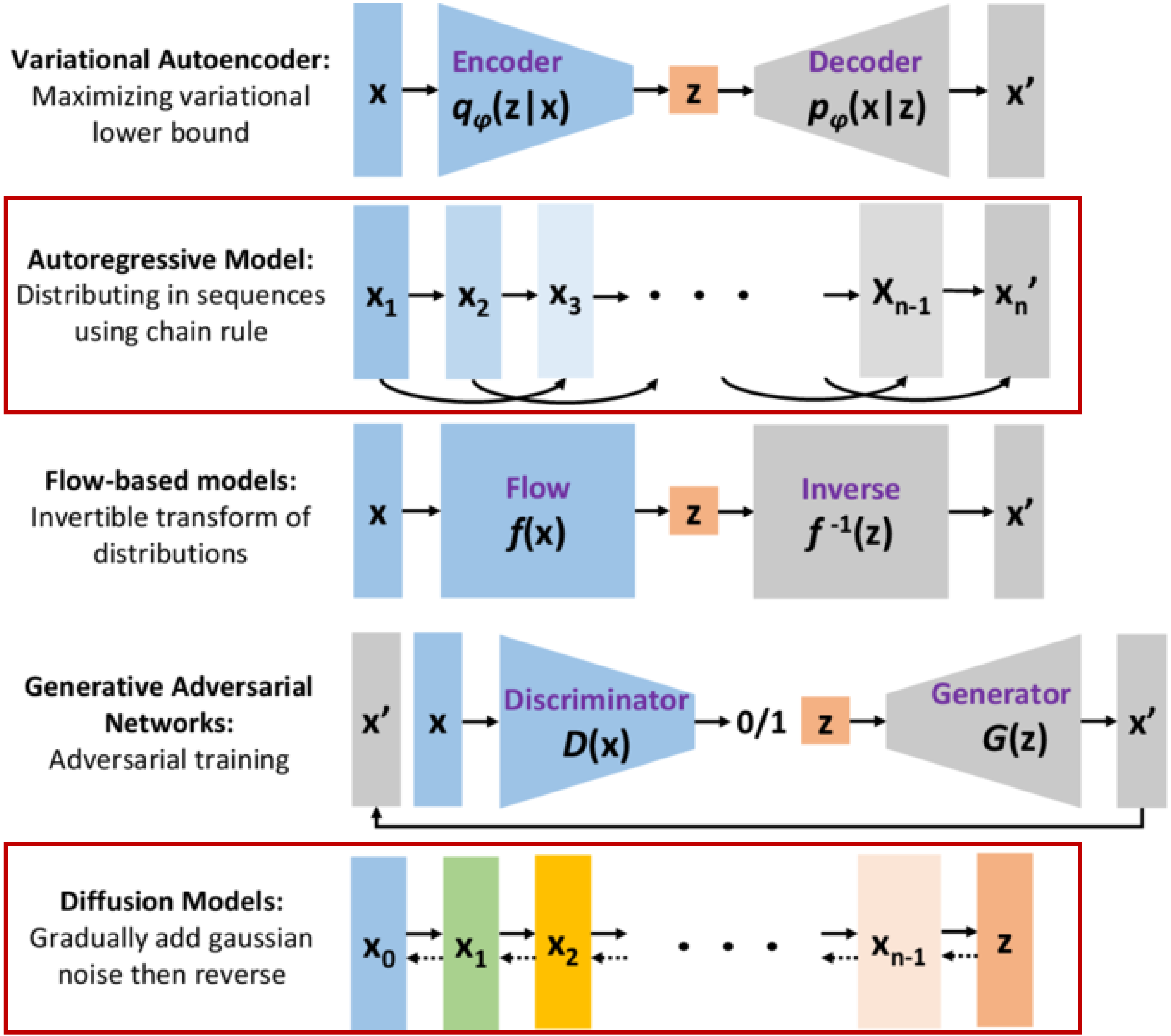
Diffusion Models



What we haven't covered about Diffusion

- Distillation (guest lecture on March 24)
- Scaling up, applications in visual generation (this Wednesday)
- The coarse-to-fine interpretation: [diffusion is spectral autoregression]
- Diffusion for discrete data (e.g., Language Modeling): [blog]





Iterative refinement

is what made GenAI really took off!

- Break the generation into many simpler steps (Divide-and-conquer)
- Provide supervision at each step
- Share model parameters across all steps

5 Minute Quiz

- On Canvas
- Passcode: pooh